סדנת תכנות בשפת C++, מס׳ קורס 67317 - 2018

תרגיל 2

תאריך הגשה: יום חמישי 3.1.19 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום שישי 4.1.19 עד שעה 23:55

תאריך ההגשה של הבוחן: יום חמישי 3.1.19 עד שעה 23:55

1. **הנחיות כלליות:**

- בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
- בכל התרגילים עליכם לכתוב קוד ברור. בכל מקרה בו הקוד שלכם אינו ברור מספיק עליכם להוסיף הערות הסבר בגוף הקוד. יש להקפיד על תיעוד (documentation) הקוד ובפרט תיעוד של כל פונקציה.
- במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
- עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
- עליכם לקמפל עם הדגלים Wall -Wextra -std=c++17 -pthread -g ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל.**
- עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקווריום, לוי, השרת river). **חובה להריץ את התרגיל במחשבי בית הספר לפני ההגשה**. (ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת bit-64 באמצעות הפקודה " uname -a" ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב x86_64)
- לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מהpresubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן על מנת שלא לאבד נקודות.

שימו לב ! תרגיל שלא יעבור את ה presubmission script ציונו ירד משמעותית (הציון יתחיל מ-50, ויוכל לרדת) **ולא יהיה ניתן לערער על כך.**

- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה.

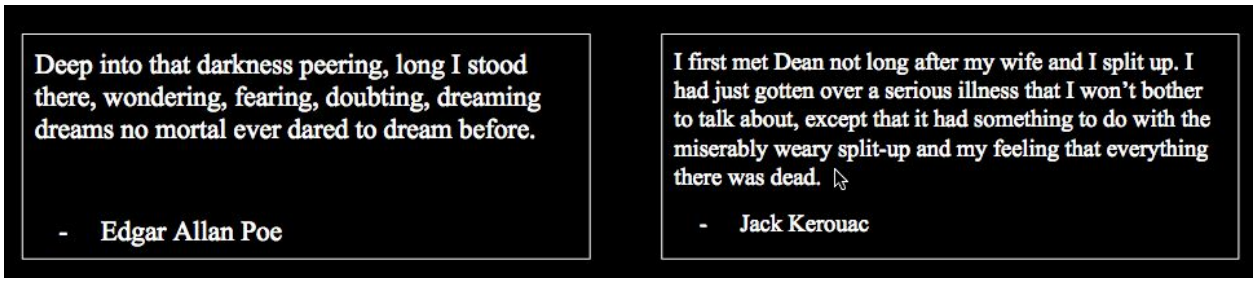
2. **הנחיות חשובות לכלל התרגילים בקורס C++**

- הקפידו להשתמש בפונקציות ואובייקטים של C++ (למשל new, delete, cout) על פני פונקציות של C (למשל malloc, free, printf).
  בפרט השתמשו במחלקה string (ב-std::string) ולא במחרוזת של C (char *).
- יש להשתמש בספריות סטנדרטיות של C++ ולא של C אלא אם כן הדבר הכרחי (וגם אז עליכם להוסיף הערה המסבירה את הסיבות לכך).
- הקפידו על עקרונות Information Hiding – לדוגמא, הקפידו כי משתני המחלקות שלכם מוגדרים כמשתנים פרטיים (private).
- הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם (היכן שניתן) by reference.
- הקפידו על שימוש במילה השמורה const בהגדרות המתודות והפרמטרים שהן מקבלות:
  המתודות שאינן משנות פרמטר מסויים – הוסיפו const לפני הגדרת הפרמטר.

מתודות של מחלקה שאינן משנות את משתני המחלקה – הוסיפו const להגדרת המתודה.
שימו לב: הגדרת משתנים / מחלקות ב- C++ כקבועים הוא אחד העקרונות החשובים בשפה.

זיהוי מחבר

בתרגיל זה נכתוב תוכנית שיודעת לזהות את המחבר של טקסט נתון לפי שכיחות של המילים הנפוצות בשפה. לכל מחבר יש סגנון כתיבה משלו. ולכן נניח שיש שכיחות דומה של מילים נפוצות בכל הטקסטים של אותו מחבר. לדוגמה:



אם המילים הנפוצות שלנו הם [I, the, there], אז מספר הופעות של מילים אלה בטקסט מימין מיוצג ע״י וקטור: [4,1,1] ובטקסט מצד שמאל [0, 0 , 1]. וקטור כזה יגדיר לנו את החתימה של המחבר. את המרחק בין שני הוקטורים נמדוד בעזרת מכפלה סקלרית:

$$\cos\theta = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

עליכם לכתוב תוכנית שמקבלת קובץ מילים נפוצות (frequent_words.txt), קובץ טקסט של מחבר לא ידוע (unknown) ומספר לא מוגבל של טקסטים של מחברים ידועים. עליכם לחשב מרחק בין ה uknown לשאר הטקסטים בעזרת חתימות של כל מחבר ובכל שורה להדפיס את שם הקובץ ו $cos(\theta)$.

<u>דוגמאות הרצה:</u>

```
> find_the_author frequent_words.txt texts/unknown.txt texts/ladygaga.txt
texts/hamilton.txt
```

התוכנית תדפיס את המרחק בין texts/unknown.txt לשני הטקסטים האחרים:

```
texts/ladygaga.txt 0.498
texts/hamilton.txt 0.984
Best matching author is texts/hamilton.txt score 0.984
```

קובץ מילים נפוצות (frequent_words.txt) הוא קובץ שבו כל שורה היא מילה.
קבצי טקסט דורשים טיפול מורכב יותר. אתם יכולים להניח שמילים מופרדות ע״י התווים הבאים: `, !:;"` כולל רווח, tab, ומעבר שורה. אין צורך לטפל בירידת שורה באמצע מילה או מקרי קצה אחרים.

<u>הערות:</u>

- דגש התרגיל הוא שימוש ב STL. לכן עליכם להשתמש במבני נתונים, איטרטורים ואלגוריתמים של STL ולכתוב כמה שפחות קוד חדש. זה מרכיב חשוב בציון תרגיל זה.
- מותר (לא חובה) להשתמש בספריות boost לניתוח (parsing) קבצי טקסט. https://theboostcpplibraries.com/boost.tokenizer
- קצת על boost כאן: (https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries
- כיוון שהטקסטים יכולים להיות ארוכים, שימו לב שעל זמן הריצה של התוכנית להיות סביר.
- אתם יכולים להתייחס למספר מילים נפוצות כקבוע. אם k זה מספר מילים נפוצות, ו N זה מספר מילים בקובץ טקסט, אז הפתרון שלכם צריך לעבוד לכל היותר ב O(Nk) לקובץ. N הרבה יותר גדול מ k.
- בתרגיל זה אתם רשאים להניח כי הקלט תקין.
- במידה ונתקלתם בשגיאה עליכם להדפיס הודעת שגיאה ל- cerr
- התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות.
- עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות.

חומר עזר: (פתרון בית ספר, קבצי קלט ו Makefile)

~labcpp/www/ex2/

הגשה:

1. עליכם להגיש קובץ tar בשם ex2.tar המכיל את הקבצים הבאים:
   - ex2.cpp with your main
   - Your additional *.h and *.cpp files
   - Makefile that builds an executable: `find_the_author`

   - שימו לב! - על אף שאתם יכולים להוסיף קבצים נוספים כרצונכם, הימנעו מהוספת קבצים לא רלוונטים (גם בכדי להקל על הבודקים, וגם בכדי שציונכם לא יפגע מכך).

2. לפני ההגשה, פתחו את הקובץ ex2.tar בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות.
3. מומלץ מאוד גם להריץ בדיקות אוטומטיות וטסטרים שכתבתם על הקוד אותו אתם עומדים להגיש. בנוסף, אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:
~labcpp/www/codingStyleCheck <file or directory>

   כאשר <directory or file> מוחלף בשם הקובץ אותו אתם רוצים לבדוק או תיקייה שיבדקו כל הקבצים הנמצאים בה (שימו לב שבדיקה אוטומטית זו הינה רק חלק מבדיקות ה codingStyle)

4. דאגו לבדוק לאחר ההגשה את קובץ הפלט (submission.pdf) וודאו שההגשה שלכם עוברת את ה- presubmission script ללא שגיאות או אזהרות.

~labcpp/www/ex2/presubmit_ex2

בהצלחה!